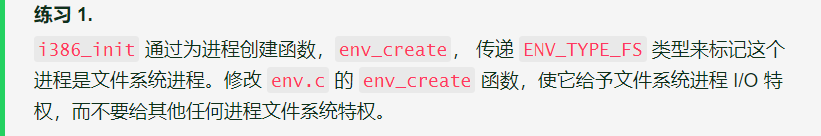
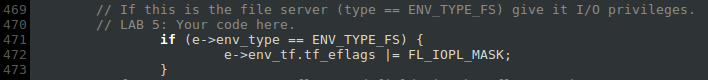
**操作系统lab5实验报告**

## .练习1

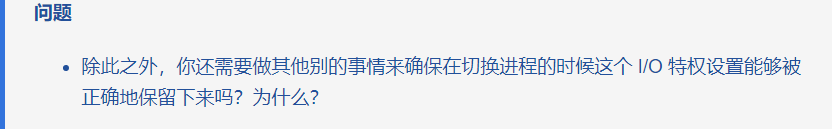


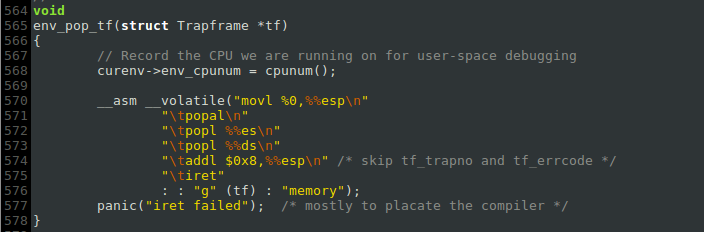
文件系统进程的type为ENV\_TYPE\_FS，需要修改env\_create()，如果type是ENV\_TYPE\_FS，需要给该进程IO权限。

在env\_create()中添加如下代码：



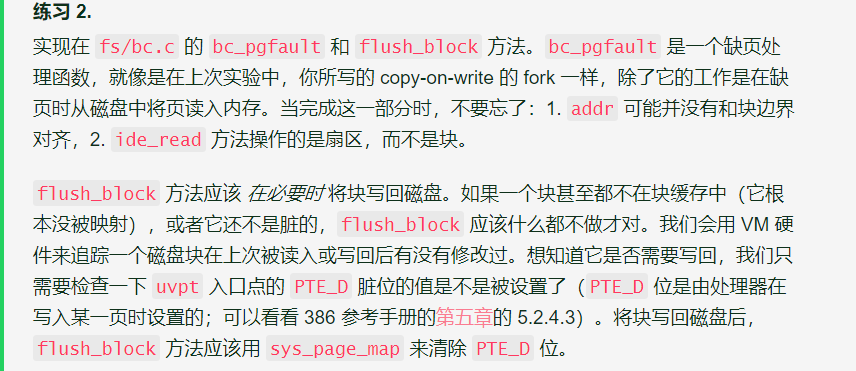
## 问题1





在进程切换时调用了env.pop\_tf函数，其中进行了寄存器的恢复，在iret指令中恢复了eip，cs，eflags等寄存器。

## 练习2

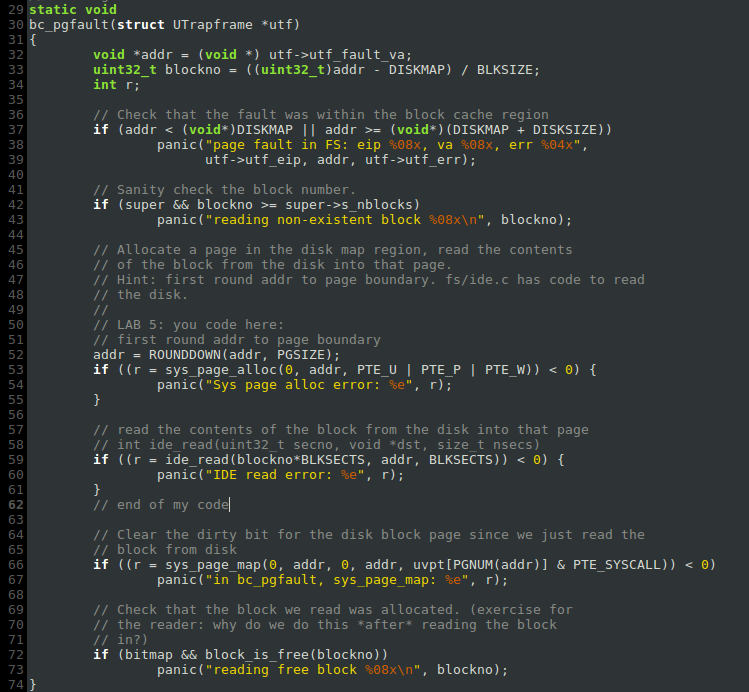


实现bc\_pgfault和flush\_block函数，其中bc\_pgfault是页错误处理程序，作用是从磁盘上装载页，flush\_block函数将一个block写入磁盘。主要用到跟磁盘直接交互的IDE驱动函数：

int ide\_read(uint32\_t secno, void \*dst, size\_t nsecs)

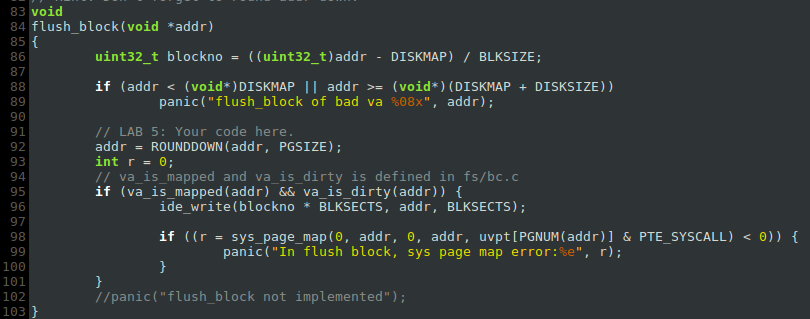
int ide\_write(uint32\_t secno, void \*dst, size\_t nsecs)

secno对应IDE磁盘上的扇区编号，dst为当前文件系统地址空间中的对应地址，nsecs为读写的扇区数。



先根据地址计算出对应的blockno，然后检查正确性包括地址是否在映射范围内、对应的block是否存在等。举例如下所示：

假设发生缺页的地址addr=10001000，块号=(addr-DISKMAP)/BLKSIZE=1，说明发生缺页在第一块，那么需要将这一块从磁盘读入内存，这一块在磁盘中对应的扇区号为：块号\*（4096/512）,每次利用ide\_read读取（4096/512）个扇区，形成一块。



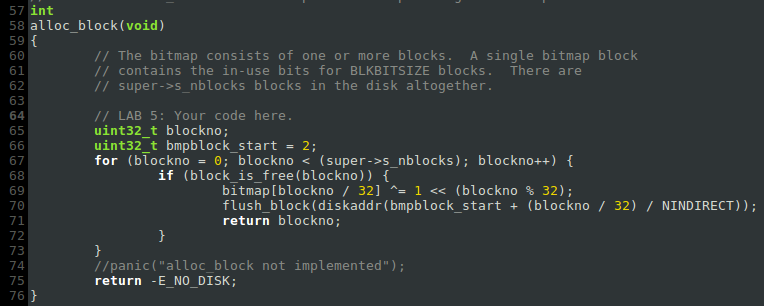
　先根据地址计算对应的blockno，然后然后检查正确性，最后判断是否是脏块并且是否被映射，如果是则写回磁盘并清除dirty位。

## 练习3



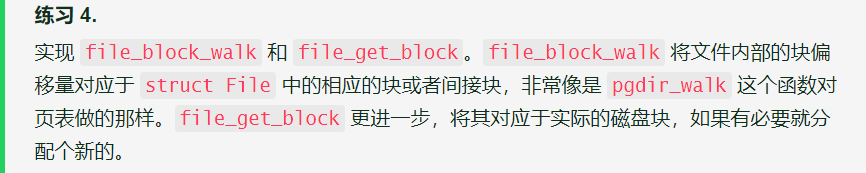
fs\_init()中已经初始化了bitmap，我们能通过bitmap访问磁盘的block 1，也就是位数组，每一位代表一个block，1表示该block未被使用，0表示已被使用（这里需要特别注意）。

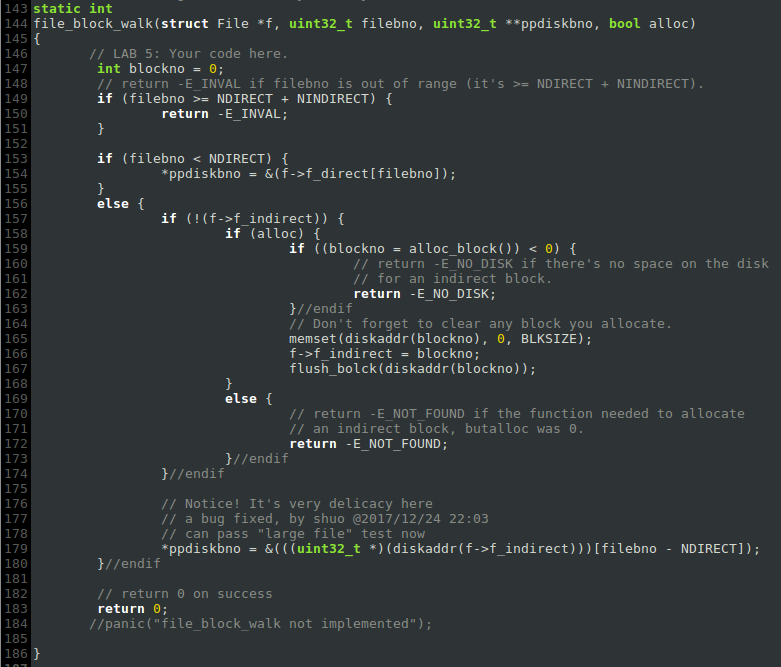
block\_is\_free，这个函数简单地检查给定的块在位图中是否被标记为空闲。

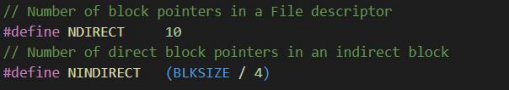


32个bit构成bitmap数组的一项，所以32个块用bitmap的一项来表示。另外，这里有一个地方需要注意，需要flush的块是从第2块到第26块中的其中一个，每一个块中有4096\*8=32K个比特，对应32K个磁盘块号，所以我们利用2+blockno/32K即可定位到需要fulsh的block。

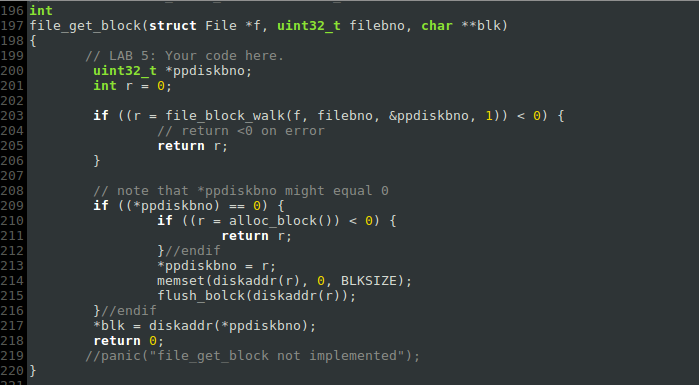
## 练习4



file\_block\_walk函数寻找一个文件结构f中的第fileno个块指向的磁盘块编号放入ppdiskbno。如果f->f\_indirect还没有分配，且alloc为真，那么将分配给新的block作为该文件的f->f\_indirect。需要注意的是如果申请了一个INDIRECT的链接块，一定要记得将其清空，并写回到磁盘中。

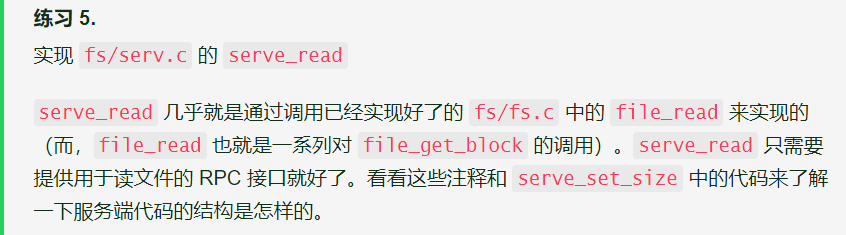


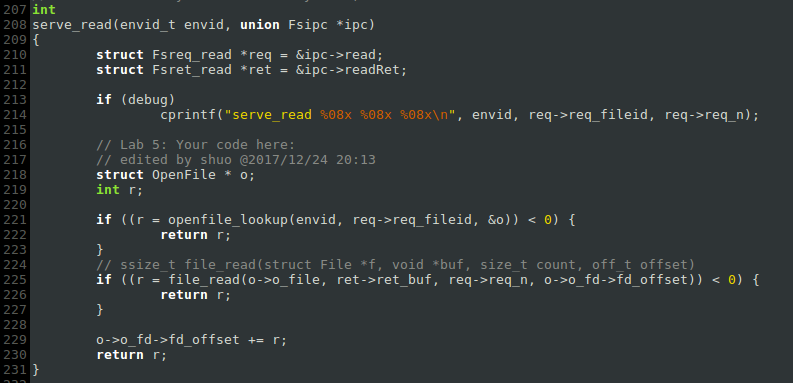
file\_get\_block函数先调用file\_walk\_block函数找到文件中的目标块，然后将其转换为地址空间中的地址赋值给blk。



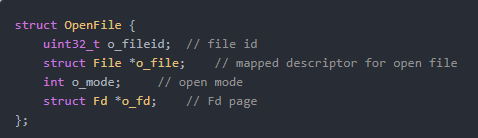
这里需要明确一个概念：File结构中无论是f\_direct还是f\_indirect，它们存储的都是指向的物理磁盘块的块号！如果要对指向的磁盘进行读写，那么必须用diskaddr转换成文件系统地址空间后才可以进行相应的操作。

## 练习5



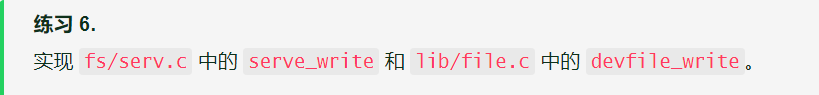


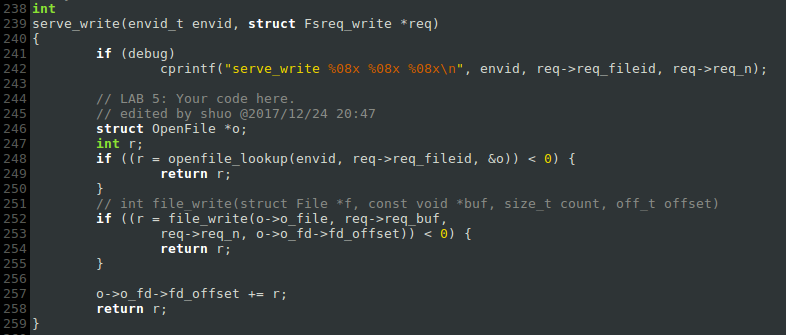
其中，openfile\_lookup的功能是根据fileid找到相应的打开文件，如果失败返回给客户端相应的错误码。接着，file\_read的功能是从f文件中查找偏移offset位置开始读取count个字节到buf中。最后移动文件偏移指针。



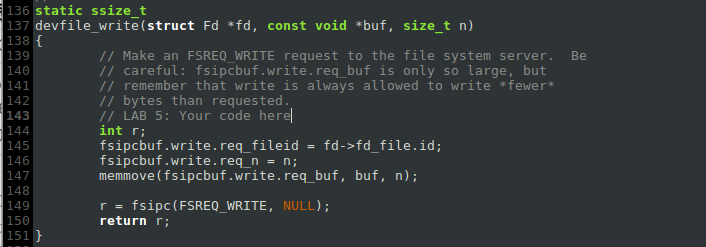
文件系统进程还维护了一个打开文件的描述符表，即opentab数组，数组元素为 struct OpenFile。OpenFile结构体用于存储打开文件信息，包括文件ID，struct File以及struct Fd。JOS同时打开的文件数一共为 MAXOPEN(1024) 个。

## 练习6





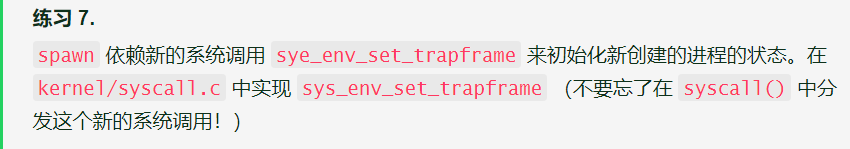
其中，file\_write的功能是从查找位置offset开始，将count个字节从buf中写入文件f。最后移动文件偏移指针。

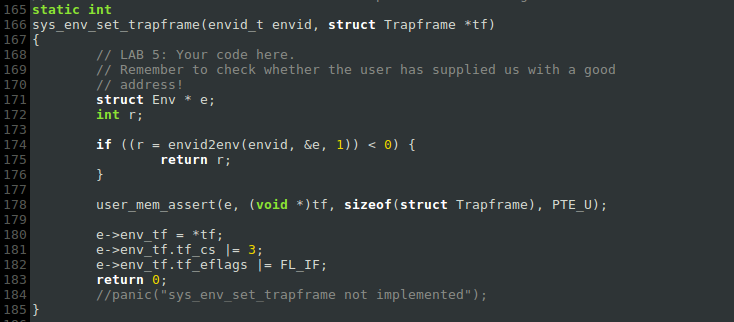


这个函数的功能是从buf到fd的当前查找位置写至多n个字节

总之：devfile\_\* 系列函数共同实现了在客户端的文件系统操作，工作方式大体上都差不多：把参数打包成请求结构体，调用 fsipc 来发送 IPC 请求，然后解包并返回结果。fsipc 方法只是简单的处理向服务器发送请求并接收回复。

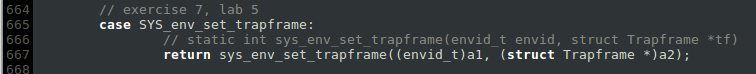
## 练习7



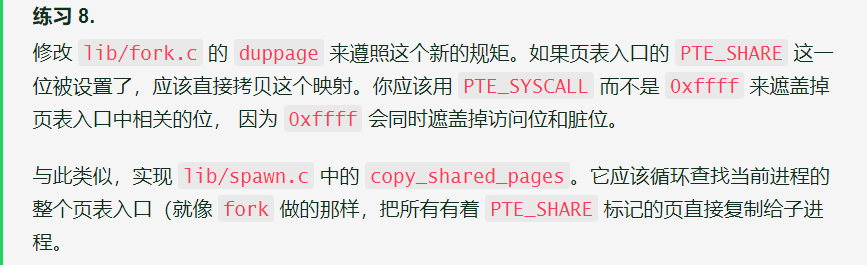


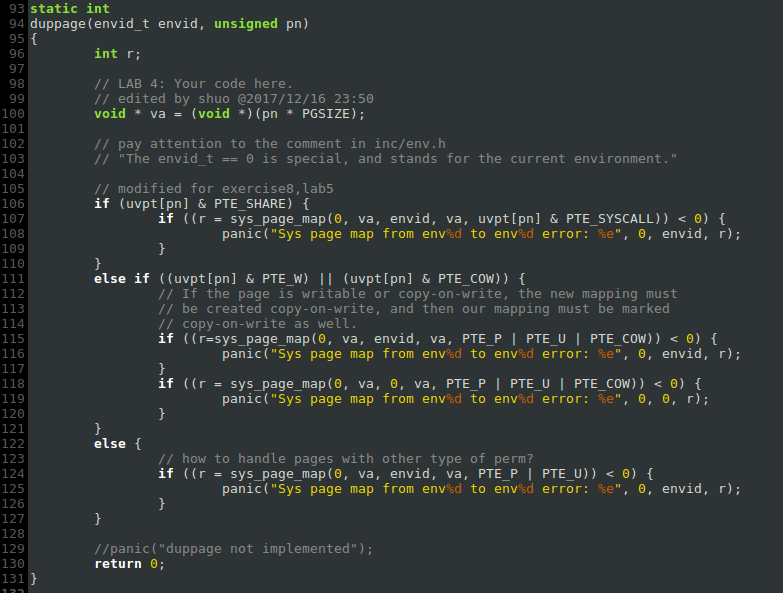
指定新进程CPL为3。

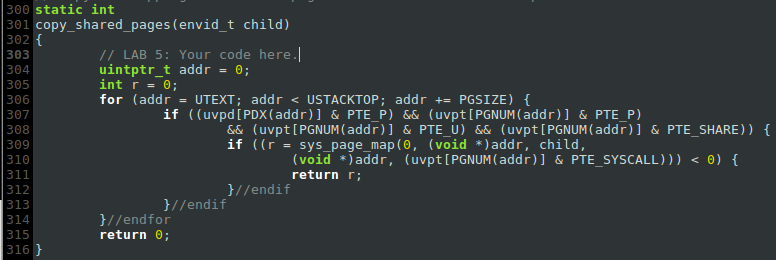
设置FL\_IF标志位打开外部中断。



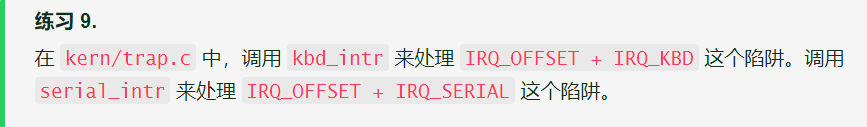
## 练习8

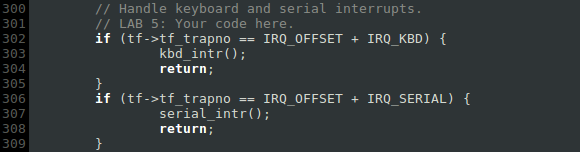






## 练习9





kbd\_intr 和 serial\_intr 会用最近读入的字符填充一个缓冲区，而控制台文件类型会从这个缓冲区中取出字符。默认情况下，控制台文件类型即作为标准输入输出，除非用户重定向了它们。

## 练习10



